

**In the claims:**

1. (currently amended) A method for automatically compiling an application program in an ~~optimum~~ a manner that optimizes one application characteristic, subject to limits on another application characteristic, comprising: compiling said application with two or more a different sets of compiler options that optimizes one application characteristic, subject to limits on another application characteristic to provide two or more executables, generating profile information from said executables, generating optimal solutions using said profile information, and automatically selecting and applying function level compiler options for said application program based upon said optimal solutions ~~profile~~ information.
2. (currently amended) The method of claim 1, wherein said two or more different sets of computer compiler options includes compile for best is ~~selected from speed, computation time, compile for best code size, and power~~ compile for intermediate points between best computation time and best code size.
3. (currently amended) A method for compiling an application program in an ~~optimum manner,~~ comprising: compiling said application program with a first set of compiler options in a manner that optimizes one application characteristic, subject to limits on another application characteristic to provide a first executable, compiling said application program with a second set of compiler options in a manner that optimizes said other application characteristic, subject to limits on said one application characteristic to provide

a second executable, generating profile information from said first and second executables, generating and displaying sets optimal solutions from said profile information wherein the sets have methods of compiling at the function level in a solution space generator, and automatically selecting and applying function level compiler options for each function of said application program based upon selected optimal solutions so as to optimize said application program ~~as a function of desired profile information.~~

4. (currently amended) The method of claim 3, wherein said first set of options is for ~~speed-~~ best computation time.
5. (currently amended) The method of claim 3 ~~4~~, wherein said second set of options is for best code size.
6. (currently amended) The method of Claim 3, further comprising the step of, analyzing said profile information against user supplied application characteristic constraints for selecting said compiler options by function.
7. (cancelled)
8. (cancelled)
9. (cancelled)
10. (currently amended) A method for compiling an application program comprising the steps of: ~~computing~~ compiling said application with a different set of compiler options that optimizes one application characteristic, subject to limits on another application characteristic to provide two or more executables; generating profile information from said executables; applying said profile information to a solver; generating sets of useful solutions from said ~~profile~~

~~information solver~~ wherein the sets have methods of for compiling at the  
~~function-level- each function;~~ and selecting ~~compiler options~~ a solution for said  
 application program using said useful solutions for subsequent compiling of  
 said application.

11. (original) The method of Claim 10 wherein said selecting step includes displaying said useful solutions.
12. (currently amended) The method of Claim 10 wherein said generating sets of useful solutions step includes generating an efficient frontier curve of optimum solution points and displaying said curve of solution points.
13. (currently amended) The method of Claim 12 wherein said generating sets of useful solutions step includes a zoom window of a section of said curve of solution points.
14. (currently amended) The method of claim 10 wherein said generating sets of useful solutions step includes linear programming and heuristics to reduce the number of permutations of option sets per function.
15. ( currently amended)The method of claim 10 wherein said generating solutions step generates possible solutions ~~step generates possible solutions~~ and filters out the possible solutions that are not better in at least one application characteristic.
16. (currently amended) The method of claim ~~10~~ 15 wherein said generating solutions step includes a search tree wherein each candidate is applied to a node and compared to the solution at the node and if faster in time and smaller in size replacing that candidate at the node, if neither faster nor smaller in size

discarding the candidate, if faster only processing down the tree in one direction and if smaller only processing down the tree in a different direction.

17. (currently amended) The method of claim 10 wherein the step of selecting a solution ~~compiling option~~ includes displaying a solution points on said solution point curve illustrating the application characteristic tradeoff that can be made by compiling for each function as prescribed by said solution ~~the compiling information of size and cycles and method of compiling.~~
18. (currently amended) The method of Claim 10 including ~~means for displaying a solution point on said solution point curve and means for displaying and overriding a compiling function solution after displaying a selected solution point and thereafter redisplaying the results~~ the solvers choice of compiler options for a particular function.
19. (currently amended) The method of Claim 17 including the step ~~after of~~ compiling of displaying by function the difference between the expected results and the actual results the application using chosen solution's set of function options.
20. (cancelled) A user interface for displaying and controlling the results of compiling an application program with a compiler having a preselected number of compiling options, comprising: a module for displaying at least a portion of solution information as a function of said selected compiling options and for selecting at least one displayed solution; and a module for outputting, for a selected solution, compiler information to allow for said application program to be compiled in a manner consistent with said selected solution.

21. (cancelled) A user interface for displaying and controlling the results of compiling an application program with a compiler having a preselected number of compiling options, comprising: a module for displaying at least a portion of information as a function of selected performance metrics and for selecting at least one displayed solution, and a module for outputting , for a selected solution, compiler information to allow for said application program to be compiled in a manner consistent with said selected solution.
22. (cancelled) The user interface of claim 21, further comprising: a module for selecting and fixing instructions for a solver that generates said solutions information.
23. (cancelled) The user interface of claim 22, further comprising: a module for compiling said application program in a manner consistent with said selected solution and for comparing the results with said selected solution.
24. (new) The method of claim 1, wherein said application characteristic includes computation time, code size and power consumption.
25. (new) The method of claim 10 wherein the step of selecting a solution includes applying a user constraint to one application characteristic and automatically selecting the solution point that meets said constraint and optimizes another application constraint.
26. (new) An apparatus for automatically generating an optimal code for an application where conflicting characteristics exist comprising:  
means for compiling said application with a compiler option to constrain a first characteristic and optimize the other characteristic and means for compiling

said application with a compiler option to constrain said other characteristic and optimize the first characteristic to provide solutions and means for automatically selecting compiler options and compiling for said application program based upon said solutions.

27.(new) The apparatus of claim 26 wherein said first characteristic is size and the other characteristic is computation time.

28. (new)The apparatus of claim 27 wherein said means for automatically selecting compiler options includes means for inputting an upper size limit for said application and said means for automatically selecting compiler options and compiling generates a solution that will minimize the computation time of the application subject to the size limit.

29.(new) The apparatus of claim 27 wherein said means for automatically selecting compiler options includes means for inputting an upper computation time limit for said application and said means for selecting compiling compiler options and compiling generates a solution that will minimize the size of the application subject to the time limit.

30. (new) An apparatus for automatically generating an optimal code for an application where conflicting characteristics exist comprising:

means for compiling said application with a compiler option to constrain a first characteristic and optimize the other characteristic;

means for compiling said application with a compiler option to constrain said other characteristic and optimize the first characteristic to provide a set of minimal solutions for the application, and

means for displaying and selecting said minimal solutions.

31. (new) The apparatus of claim 30 wherein each solution in the set of minimal solutions is such that the apparatus cannot compile an alternate solution that is both smaller in size and in computation time.

32 (new) A method for automatically generating an optimal code for an application comprising:

compiling, measuring and recording size and computation time of the functions in an application while varying the compiler options to get size versus computation time behavior data for each function with compiled with a set of compiler options, and means for generating a set of minimal solutions for the application such that the compiler cannot generate an alternate solution that is both smaller in size and in computation time.

33. (new) The method of claim 32 wherein said means for generating a set of minimal solutions includes the steps of linear programming and heuristics to remove non-minimal solutions.